

# Animatable 3D Gaussian: Fast and High-Quality Reconstruction of Multiple Human Avatars

Anonymous Authors

## A OVERVIEW

In the supplementary material, we introduce the calculation process of bone transformation and discuss the feasibility of optimizing the joint positions in canonical space. Furthermore, we provide more visualization results. We encourage the reader to watch the video for our high-quality results.

## B CALCULATION OF BONE TRANSFORMATION

We introduce bone transformation in Sec. 3.2. In this section, we provide the detailed calculation process of Eq. 7 and discuss the need to optimize the joint positions in canonical space.

We combine posed translation  $T_t$  and rotation angle  $\omega_1^t$  to obtain the posed transformation  $B_{posed,1}^t$  of the root joint:

$$B_{posed,1}^t = \begin{bmatrix} \mathbf{R}(\omega_1^t) & T_t \\ 0 & 1 \end{bmatrix}, \quad (20)$$

where  $\mathbf{R}(\cdot)$  denotes a function that converts the Euler angle to a rotation matrix.

Similarly, we combine the local positions  $J_i$  and rotation angles  $\omega_i^t$  of the remaining joints to obtain their local posed transformations:

$$B_{local\_posed,i}^t = \begin{bmatrix} \mathbf{R}(\omega_i^t) & J_i \\ 0 & 1 \end{bmatrix}. \quad (21)$$

Next, we calculate the posed transformation for each joint in turn according to the connection order:

$$B_{posed,i}^t = B_{local\_posed,i}^t B_{posed,parent(i)}^t, \quad (22)$$

where  $parent(i)$  refers to the parent joint index of joint  $i$ . The calculation starts from the root joint.

In order to obtain the bone transformation from canonical space to posed space, we need the canonical transformations of joints. To simplify the calculation, we assume that the local rotation matrix of the canonical joints is the identity matrix, which can be achieved by preprocessing. We first calculate the local canonical transformations as follows:

$$B_{local\_can,i}^t = \begin{bmatrix} I & J_i \\ 0 & 1 \end{bmatrix}. \quad (23)$$

As with Eq. 22, we compute the canonical transformation for each joint in turn:

$$B_{can,i}^t = B_{local\_can,i}^t B_{can,parent(i)}^t. \quad (24)$$

Finally, we obtain the bone transformation mentioned in Eq. 7 by multiplying the inverse canonical transformation and the posed transformation:

$$B_i^t = B_{posed,i}^t \cdot (B_{can,i}^t)^{-1}. \quad (25)$$

Through the above calculation, we realized the mapping from the canonical joint positions  $\mathbf{J}$ , the posed rotation angle  $\mathbf{S}_t$  and the posed root translation  $T_t$  to the bone transformation  $\mathbf{B}_t$ . In other words, pose-guided deformation only relates to the joint position in canonical space. Therefore, we can implement different

deformations for different humans by optimizing joint positions in canonical space.

## C IMPLEMENTATION DETAILS

We extend the CUDA kernels of the 3D Gaussian rasterizer to achieve 3D Gaussian deformation and use the tiny-cuda-nn to implement the hash-encoded parameter field.

**Network Architecture.** To facilitate expansion, we build a hash-encoded network for each parameter. For spherical harmonic coefficients and vertex displacement, we use a hash table of length  $2^{17}$  and a multi-layer perceptron with two hidden layers of 64 nodes each. For time-dependent ambient occlusion, we use a hash table of length  $2^{19}$  and a multi-layer perceptron with two hidden layers of 64 nodes each.

**Training Details.** Since our method requires rapid convergence over several epochs, we set a fixed learning rate for each parameter. We use second-order spherical harmonics and optimize all coefficients at the beginning to speed up convergence. In order to learn both time-independent spherical harmonic colors and time-dependent ambient occlusion, we do not enable ambient occlusion until the fifth epoch.